The 3rd International Symposium on Engineering, Energy and Environments
17-20 November 2013, Pullman King Power Hotel, Bangkok

# A Python Project for Lagrangian Mechanics

Peter Slaets[1*], Pauwel Goethals[1†], Dries Haeseldonckx[1‡], Sietze Swolfs[2§], and Maarten Vanierschot[1¶]

[1]Assistent professor at the unit Energy, Group T - International University College Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium
[2]Research assistent at the unit Energy, Group T - International University College Leuven, Andreas Vesaliusstraat 13, 3000 Leuven, Belgium

**Abstract**

Climate change mitigation imposes a world-wide technological challenge for both government and industry. The European commission translated this challenge into an Energy Efficiency Directive that establishes a common framework of measures to ensure 20% power reduction in 2020. The industrial sector is responsible for around 24 % of the total energy consumption. Within the industrial sector the manufacturing industry accounts for a large fraction of this energy consumption. Technological advances like advanced control or mechanical redesign can significantly improve the energy efficiency of many manufacturing processes. This paper presents a platform independent, open source software application that enables end-users to specify a dynamic model and analyse its response. The system dynamics are specified using Lagrangian mechanics. The elegance of Lagrangian mechanics resides in the straightforward design procedure based on the Lagrangian. However, conversion of the Lagrangian to a dynamic model is computational intensive and often impossible to perform manually. These computations are performed by the symbolic library 'Sympy' written in Python, a popular platform independent general-purpose, high-level programming language. Simulation results of common mechanical systems are shown that illustrate the user-friendly design interface and the straightforward design procedure.

*Keywords*: Python, Lagrangian mechanics, open source.

## 1 Introduction

Reduction of energy consumption is an important issue all over the world because of rising energy prices and the increased attention for their direct or in-direct environmental pollution.

*Peter Slaets, email:<firstname>.<lastname>@groept.be,tel:+3216301075, fax:+3216301040
†Pauwel Goethals, email:<firstname>.<lastname>@groept.be,tel:+3216301074, fax:+3216301040
‡Dries Haeseldonckx, email:<firstname>.<lastname>@groept.be,tel:+3216301071, fax:+3216301040
§Sietze Swolfs, email:<firstname>.<lastname>@groept.be,tel:+3216301071, fax:+3216301040
¶Maarten Vanierschot, email:<firstname>.<lastname>@groept.be,tel:+3216301071, fax:+3216301040

The European commission imposed an Energy Efficiency Directive [1] that establishes a common framework of measures to ensure 20% power reduction in 2020. The European industrial sector is responsible for around 24 % of the total energy consumption. Within the industrial sector the manufacturing industry accounts for a large fraction of this energy consumption.

Technological advances in the field of energy monitoring , energy management systems and energy efficient actuation can significantly improve the energy efficiency of many manufacturing processes. Effort have been done to exploit inherent dynamical properties of the dynamical system to build up a motion planning framework [2, 3]. Others try to improve the motion trajectories of industrial machinery e.g. machine tools [4] or industrial robots [6]. In all theses cases an analytic dynamic model of the system is required.

This paper presents a platform independent open source software application that simplifies the build-up and simulation of a mechanical dynamic model. The paper is build up as follows: In a first section Lagrangian mechanics [7] is explained and an elegant design procedure for Lagrangian mechanics is proposed. However, conversion from the Lagrangian to a dynamic model is computational intensive and often impossible to perform manually. Therefore a Python software GUI application based on the symbolic library SymPy [5] is used to perform these computations. The software application is free to use, even for commercial products, and is distributed using a open source GPL license. In the last section the design procedure is applied to two common mechanical systems i.e. the double pendulum and a dual driven electric vehicle. The results shown illustrate the user-friendly design interface and the straightforward design procedure.

## 2 Lagrangian mechanics

Lagrangian mechanics is build up around a quantity called the **Lagrangian**. Consider a system with $m$ motion degrees of freedom, denoted by the m-dimensional generalized coordinate vector $\mathbf{q}$[1]$= [q_1, \ \ldots \ , q_m]^T$. The kinetic energy generally depends on both the coordinate and velocity $\dot{\mathbf{q}}$[2], so we write it as $T(\mathbf{q}, \dot{\mathbf{q}})$. Similarly, we write the potential energy of the system as $U(\mathbf{q}, \dot{\mathbf{q}})$.

The Lagrangian $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ is specified by the equation

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}, \dot{\mathbf{q}}) \tag{1}$$

where $\dot{\mathbf{q}}$ represents the vector with generalized velocities.

The system dynamics are fully specified by the equation of motion. This is given by the 2nd kind of Lagrange equations based on a set of $m$ differential equations:

$$\frac{d}{dt}\frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})}{\partial \dot{q}_j} = \frac{\partial \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})}{\partial q_j} + F_{q_j} \tag{2}$$

where $j = 1, 2, ...m$ represents the j-th degree of freedom with corresponding generalized coordinate $q_j$, and generalized velocity $\dot{q}_j$. The generalized force for each degree of freedom $F_{q_j}$ is chosen such that the product $F_{q_j}\delta q_j$ quantifies the work done by the driving forces when

[1] All vectors are indicated in bold

[2] The dot on top of a symbol refers to the time derivative $\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt}$

$q_j$ is changed by $\delta q_j$.

The procedure to obtain the dynamic motion equations is:

1. Specify the kinetic energy $T(\mathbf{q}, \dot{\mathbf{q}})$, and the potential energy $V(\mathbf{q}, \dot{\mathbf{q}})$ and compute the Lagrangian $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}, \dot{\mathbf{q}})$.
2. Compute the partial derivative $\frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial q_j}$ for all degrees of freedom $j = 1, 2, ...m$.
3. Compute $\frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{q}_j}$ and derive $\frac{d}{dt}\frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{q}_j}$ for all degrees of freedom $j = 1, 2, ...m$ . It is important that $\dot{q}_j$ is treated as a complete variable in its own right, and not as a derivative.
4. Write down $\frac{d}{dt}\frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{q}_j} = \frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial q_j}$ for all degrees of freedom $j = 1, 2, ...m$ . These are the Euler-Lagrange equations.
5. Rewrite the set of differential equations obtained in the previous step as a set of Ordinary Differential Equations (ODE's).

$$\ddot{q}_1 = f(\dot{q}_2, \ldots, \dot{q}_m, F_{q_1}) \quad (3)$$

$$\vdots \quad (4)$$

$$\ddot{q}_m = f(\dot{q}_1, \ldots, \dot{q}_{m-1}, , F_{q_m})$$

At this point, $\dot{q}_j$ is treated as a normal variable. Note that in general the system dynamics consists of a set of $m$ ODE's.

# 3 Applications

In this section the dynamic model equations of two common mechanical systems, i.e. a double pendulum and a dual drive vehicle, are derived using the Lagrangian Python GUI and a simulation is performed. Both examples illustrate the modelling and simulation capabilities of the designed Python application. An installation packages for various platform (Unix, Windows, Mac) can be found at `http://code.google.com/p/euler-lagrange-dynamics/`.

## 3.1 Double pendulum

A double pendulum consists of one pendulum attached to another. Consider a double bob pendulum with masses $m_1$ and $m_2$ attached by rigid massless wires of lengths $l_1$ and $l_2$. Further, let the angles the two wires make with the vertical be denoted by $\theta_1$ and $\theta_2$, as illustrated in figure 1. Finally, let gravity be given by g. Then the positions of the bobs are specified by

$$x_1 = l_1 \sin(\theta_1) \quad (5)$$

$$y_1 = -l_1 \cos(\theta_1) \quad (6)$$

$$x_2 = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) \quad (7)$$

$$y_2 = -l_1 \cos(\theta_1) - l_2 \cos(\theta_2) \quad (8)$$

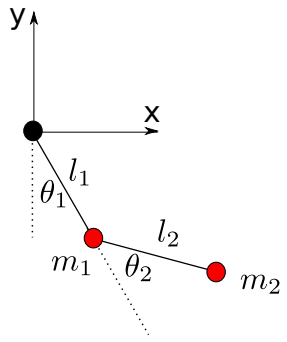


Figure 1: A double pendulum.

The potential energy of the system is then given by

$$U(\mathbf{q}, \dot{\mathbf{q}}) \quad = \quad m_1 g y_1 + m_2 g y_2 \qquad (9)$$

and the kinetic energy by

$$T(\mathbf{q}, \dot{\mathbf{q}}) \quad = \quad m_1 (v_1)^2/2 + m_2 (v_2)^2/2 \qquad (10)$$

The previously specified kinetic and potential energy are inserted in the Python GUI as shown in figure!2.

| | |
|---|---|
| Degrees of freedom: q1,q2, ... | q1,q2 |
| Parameters: p1,p2,... | m1,m2,l1,l2,g |
| Kinetic energy | 0.5*m1*pow(l1*diff(q1(t),t),2)+ 0.5*m2* pow(l1*diff(q1(t),t),2) + 0.5*m2*pow(l2*diff(q2(t),t),2) + m2*l1*l2*diff(q2(t),t)*diff(q1(t),t)*cos(q1(t)-q2(t)) |
| Potential energy | (-m1-m2)*g*l1*cos(q1(t)) - m2*g*l2*cos(q2(t)) |
| Generalized forces: F1;F2;... | 0;0 |
| Initial values DOF:4,5,5, ... | 0,0,1,0 |
| Values for parameters: 1,2,... | 1,1,1,1,9.8 |
| Simulation Time [sec] | 20 |

Figure 2: The python GUI for a double pendulum with degrees of freedom $q_1 = \theta_1, q_2 = \theta_2$ and parameters $m1, m2, l1, l2, g$ and corresponding initial value $(q_1(0) = 0, q_2(0) = 0, \dot{q}_1(0) = 0, \dot{q}_2(0) = 0)$ and $(m1 = 1, m2 = 1, l1 = 1, l2 = 1, g = 9.8)$. To specify the Lagrangian the kinetic en potential energy function are given together with the generalized force and simulation time.

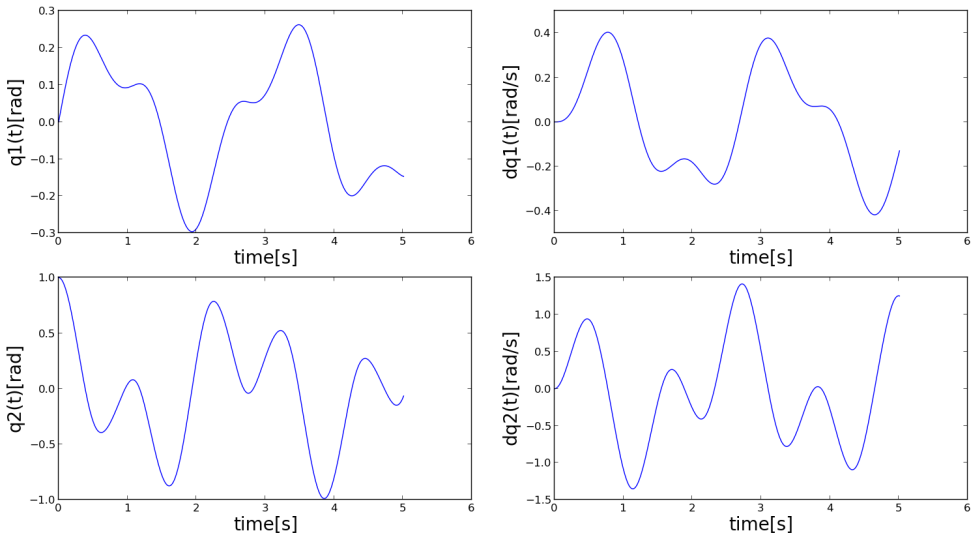


Figure 3: The simulation results for the position and velocity of both degrees of freedom $q1 = \theta_1, q2 = \theta_2$ resulting from the input given in figure 2.

### 3.2 Dual driven vehicle

A dual driven vehicles is shown in figure 4. The left and right back-wheel are driven separately by two motors. The wheels have a radius $r$, with inertia $J_w$. These wheels are positioned on a distance $b$ from the centre of the vehicle. $J_c$ represents the inertia of the vehicle for rotation in the horizontal plane about its centre of mass. The angular position of the right and left wheel respectively are given by $\theta_R$ and $\theta_L$. The vehicle has a mass $m$. The relationship between right and left wheel angular velocity $\omega_R, \omega_L$ and vehicle linear and rotational velocity $v, \omega$ is:

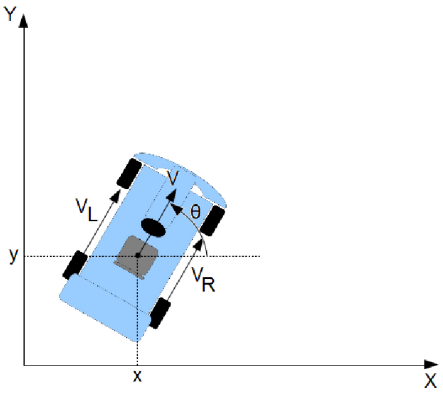


Figure 4: A dual driven vehicle.

$$
\begin{aligned}
v_L &= r \cdot \omega_L, \\
v_R &= r \cdot \omega_R, \\
\omega &= \frac{v_R - v_L}{2b}, \\
v &= \frac{v_R + v_L}{2}.
\end{aligned}
$$

The non-linear Cartesian kinematic equations in $v$, $\omega$ and $\theta$ are:

$$
\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{11}
$$

The potential energy is equal to zero because we assume that there is no vertical displacement and the kinetic energy is given by

$$
T(\mathbf{q}, \dot{\mathbf{q}}) \quad = \quad \frac{1}{2}mv^2 + \frac{1}{2}J_c\dot{\theta}^2 + \frac{1}{2}J_w(\omega_R^2 + \dot{\omega}_L^2)
$$

The previously specified kinetic and potential energy are inserted in the Python GUI.

| | |
|---|---|
| Degrees of freedom: q1,q2, ... | q1,q2 |
| Parameters: p1,p2,... | m,Jc,Jw,R,b |
| Kinetic energy | 0.125*Jc*(R*Derivative(q1(t), t) - R*Derivative(q2(t), t))**2/b**2 + 0.5*Jw*(Derivative(q1(t), t)**2 + Derivative(q2(t), t)**2) + 0.5*m*(R*Derivative(q1(t), t)/2 + R*Derivative(q2(t), t)/2)**2 |
| Potential energy | 0 |
| Generalized forces: F1;F2;... | cos(t);sin(t) |
| Initial values DOF:4,5,5, ... | 0,0,0,0 |
| Values for parameters: 1,2,... | 10,0.1,0.01,0.2,0.3 |
| Simulation Time [sec] | 20 |

Figure 5: The python GUI filled in for a dual driven vehicle with degrees of freedom $q_1 = \omega_L, q_2 = \omega_R$ and parameters $m, Jc, Jw, R, b$ and corresponding initial value $(q_1(0) = 0, q_2(0) = 0, \dot{q}_1(0) = 0, \dot{q}_2(0) = 0)$ and $(m = 10, Jc = 0.1, Jw = 0.01, R = 0.2, b = 0.3)$. To specify the Lagrangian the kinetic en potential energy function are given together with the generalized force $(\cos(t); \sin(t))$ and simulation time $(T = 20$ sec).

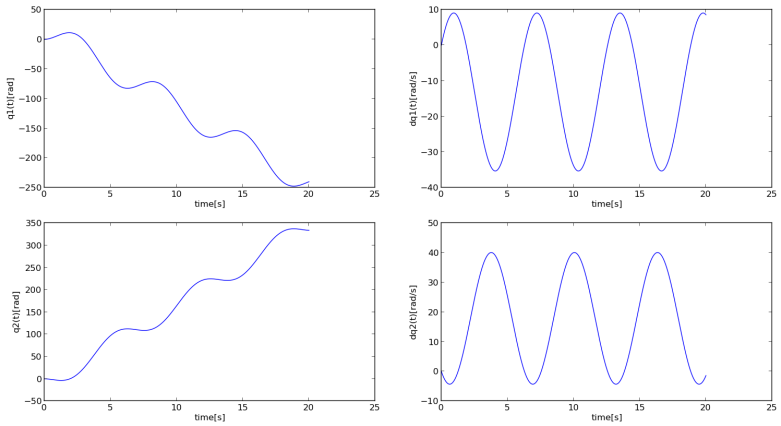


Figure 6: This figure shows the simulation results for the position and velocity of both degrees of freedom $q1 = \omega_L, q2 = \omega_R$ resulting from the input given in figure 5.

## 4 Conclusions

This paper introduces a Python based simulation application for Lagrangian mechanics applied to two well-known mechanical systems. The resulting GUI generates the symbolic dynamic model equations based on the user input and performs a numerical simulation. Future work will focus on the extension of the package to Hamiltonian mechanics to directly obtain a set of first order differential equations.

## References

[1] European Parliament and Council. Directive 2012/27/EU of the European Parliament and of the Council of 25 October 2012 on energy efficiency. ISSN 1977-0677, October 2012.

[2] K. Flaskamp and S. Ober-Blobaum. Energy efficient control for mechanical systems based on inherent dynamical structures. In *American Control Conference (ACC), 2012*, pages 2609–2614, 2012.

[3] P. Slaets, J. Janssenswillen, and S. Swinnen, R. Swolfs. Optimal Dynamic Predictive Cruise Control for differential driven electric vehicle. In *2012 EVS26 Online Conference Proceedings, Los Angeles, USA*. EDTA Electric Drive Transportation Association, may 2012.

[4] V. Smirnov, V. Plyusnin, and G. Mirzaeva. Energy efficient trajectories of industrial machine tools with parallel kinematics. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1267–1272, 2013.

[5] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2008.

[6] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-optimal path tracking for robots: A convex optimization approach. *Automatic Control, IEEE Transactions on*, 54(10):2318–2327, 2009.

[7] D. Wells. *Schaum's Outline of Lagrangian Dynamics*. Schaum's Outline Series. McGraw-Hill Education, 1967.